#27

NOT ENTERED See paper No 30

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

APPLICATION OF MARTIN G. REIFFIN	
FOR: COMPUTER SYSTEM WITH REAL-TIME CODE PROCESSING	ART UNIT 232
FILED: APRIL 3, 1985	EXAMINER T. LEE
SERIAL NO. 719,507	

REPLY BY APPELLANT

This is appellant's reply to the Examiner's Answer dated December 1, 1988 in the above appeal. The Answer stated at least one new ground of rejection. This reply includes an accompanying amendment incorporated herein by reference and adding the following new claims to the present application:

NEW CLAIMS

57. A computer system as recited in Claim 51 wherein said character codes constitute a language having rules and said code processor program is a language processor comprising

means for analyzing the character codes in the buffer for conformity with at least a subset of the rules of the language.

58. A computer system as recited in Claim 57 wherein said language processor comprises

means for emitting an error message in response to an instance of nonconformity with a rule of the language.

59. A computer system as recited in Claim 57 wherein said analyzing means comprises

lexical analyzer means for determining conformity with the spelling rules of the language, and

syntactic analyzer means for determining conformity with the grammatical rules of the language.

60. A computer system as recited in Claim 54 wherein said character codes constitute a language and said code processor program is a language processor comprising

means for analyzing the character codes in the buffer for conformity with the language.

61. A computer system as recited in Claim 60 wherein said language processor comprises

means for emitting a message in response to an instance of nonconformity with the language.

62. A computer system as recited in Claim 60 wherein said analyzing means comprises

lexical analyzer means for determining conformity with the spelling rules of the language.

63. A computer system as recited in Claim 60 wherein said analyzing means comprises

syntactic analyzer means for determining conformity with the grammatical rules of the language.

64. A computer system as recited in Claim 56 wherein said character codes constitute a language having rules and said code processor program is a language processor comprising

means for analyzing the character codes in the buffer for conformity with the rules of the language.

65. A computer system as recited in Claim 64 wherein said language processor comprises

means for emitting an error message in response to an instance of nonconformity with a rule of the language.

66. A computer system as recited in Claim 64 wherein said analyzing means comprises

means for determining conformity with the spelling rules of the language.

67. A computer system as recited in Claim 64 wherein said analyzing means comprises

means for determining conformity with the grammatical rules of the language.

NEW POINTS OF ARGUMENT RAISED IN THE EXAMINER'S ANSWER

Since Claims 51, 54 and 56, the original claims on appeal, were submitted after the final rejection, the Examiner's Answer raised new points of argument not raised in the final rejection. These new points are grouped and discussed below under the following broad headings stated as appellant's contentions:

I. THE LAWRENCE ET AL. AND MADDOCK TEXT EDITORS ARE NOT INTERRUPT SERVICE ROUTINES WHICH GET CONTROL IMMEDIATELY UPON STRIKING A KEY

The Lawrence et al. text editor is neither an interrupt service routine nor does it get control of the central processing unit immediately upon striking a key.

Appellant's Claim 51 recites:

"means responsive to striking one of said keys to activate said interrupt input whereby the contents of the program counter are pushed onto the stack and the memory address of the interrupt service routine is loaded into the program counter thereby immediately passing control of the central processing unit to the interrupt service routine, whereupon the code processor program is interrupted immediately upon completing the machine instruction which it was executing when the key was struck,

said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key,..."

Appellant's Claim 54 recites:

"means responsive to striking one of said keys to activate said interrupt input to interrupt the code processor program and pass control of the central processing unit to the interrupt service routine,

said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key,"

Apparently persuaded by Paragraph 14 of the third Reiffin affidavit that an editor which is "invoked" is not an "interrupt service routine", Paragraph 21 of the Examiner's Answer asserts that the Lawrence disclosure contained:

"...some inconsistency of the terminology, i.e. interrupted versus invoked."

Paragraph 53 of the Examiner's Answer asserts falsely:

"53. Appellant pointed out the inconsistence [sic] of terms, i.e. interrupted and invoked, used in the reference. Appellant alleged that the inconsistency should be resolved in favor of the term 'invoked', not 'interrupted'."

This assertion is untrue. Appellant did not point out any "inconsistence of terms" in the Lawrence disclosure and did not concede that there was any "inconsistence" to be "resolved".

It is not the Lawrence disclosure which is inconsistent, but rather the Examiner's interpretation of the disclosure is inconsistent with the actual disclosure.

The Examiner's Answer wrestles vainly with the numerous inconsistencies between the Examiner's version and the disclosed version of the Lawrence system. For example:

Paragraph 21 of the Examiner's Answer states:

"Lawrence et al taught that the formatting means was invoked by the editing means to perform formatting operation."

Paragraph 22 of the Examiner's Answer criticizes Lawrence for disclosing a different mode of operation than the Examiner prefers, as follows:

> "The interrupted program [formatting means] should not have been invoked by the second program [editing means]."

Paragraph 23 of the Examiner's Answer then concludes, completely contrary to both his own statement in Paragraph 21 and the clear disclosure in the Lawrence reference, that:

"...the formatter was interrupted by editor wherein the control was subsequently returned, not invoked by the editor, to the formatter."

In support of this remarkable conclusion, the Examiner cites Col. 11, Lines 51-56 of the Lawrence specification which completely refutes the Examiner's contention by stating:

"If a key is depressed during formatting, then the interpreter is terminated at the end of the current row and control is passed back to the keystroke processor 9." (Emphasis added.)

As explained in Paragraphs 13 and 16 of appellant's third affidavit, this cited sentence is inconsistent with the

Examiner's contention in three respects:

1. An interrupted routine is not "terminated" (Third Reiffin affidavit, Par. 13);

2. An interrupt takes effect immediately and does not permit any delay allowing the interrupted routine to continue control of the central processing unit until it reaches "the end of the current row" (Third Reiffin affidavit, Par. 16); and

3. The cited Lawrence statement asserts that control is passed to "the keystroke processor 9", and not to the text editor. The text editor gets control only after the keystroke processor "invokes" it. (Col. 3, Lines 8,9)

Both the abstract and specification of the reference patent consistently and repeatedly make assertions contrary to the Examiner's contention that the Lawrence editor gets control as an interrupt service routine which immediately "interrupts" the interpreter/formatter to which control then automatically "returns". These contrary assertions include the following:

(1) "An interpreter/formatter is invoked..." (Abstract, Lines 10,11):

As recognized by the Examiner in Paragraphs 21 and 22 of the Examiner's Answer, the term "invoked" is inconsistent with the formatter automatically resuming control upon return from an interrupt (Third Reiffin affidavit, Par. 15).

(2) "Formatting is inhibited by the editor if it receives a keystroke ..." (Abstract, Lines 17,18):

The term "inhibited" is not used in the art as a synonym for "interrupted".

(3) "... each extended formatting command consists of the

CSP followed by a class byte, a count byte, a type byte, and, optionally, parameters." (Col., Lines 49-51):

Four or more bytes are required before formatting can be performed after an insertion of an extended formatting command. Turning control over to the formatter after each and every keystroke would be pointless.

(4) "A keystroke processor (K/SP) 9 ... invokes a text editor (TE) 10." (Col. 3, Lines 8,9):

Here again the term "invokes" is admitted by the Examiner to be inconsistent with "interrupts", and is so in fact (Third Reiffin affidavit, Par. 14,15).

(5) "Processing by the interpreter/formatter 6 is <u>initiated</u> either by the segmenter 5 on completion of transferring incoming text to the store 2 or by the text editor 10 on completion of an update to the text in the store 2." (Col 3, Lines 25-28) (Emphasis added.):

An allegedly interrupted routine (interpreter/formatter) cannot be said to be "initiated" by an alleged interrupt service routine (text editor 10). (Third Reiffin affidavit, Par. 13-15).

(6) "The text editor 10 employs a two-step process. Firstly, it invokes the interpreter/formatter 6 to determine the physical cursor position ... Secondly, the particular operation on the text is performed (insertion/deletion) at the logical cursor position and the interpreter/formatter is re-invoked to display the effect of the change on the screen." (Col. 3, Lines 29-38):

As apparently admitted in Paragraphs 21 and 22 of the Examiner's Answer, an alleged interrupt service routine (text editor 10) cannot be said to "invoke" an allegedly interrupted routine (interpreter/formatter), nor can an allegedly interrupted routine be said to resume control by being "re-invoked" (Third Reiffin affidavit, Par. 14,15).

(7) "Data or text to be displayed is loaded into section 8 from section 35 by the microprocessor when the data or text is called by the operator or a program." (Col. 6, last line to Col. 7, Line 3):

Section 8 is the refresh buffer. Loading into Section 8 is the formatting process. Apparently the formatter gets control only "when called by the operator or a program", and not upon each return from a keystroke interrupt as contended by the Examiner.

(8) "Formatting starts at the top of the screen ..." (Col. 10, Lines 67,68; Col. 11, Line 5; Col. 11, Line 12):

Therefore the formatter does not resume processing at the point where it was "interrupted", and does not automatically resume control upon return from an interrupt.

(9) "If a key is depressed during formatting, then the interpreter is terminated at the end of the current row and control is passed back to the keystroke processor 9." (Col. 11, Lines 51-54):

The word "terminated" is inconsistent with "interrupted", as explained in Paragraph 13 of the third Reiffin affidavit. Also, an interrupt service routine gets control immediately, and transfer of control cannot be delayed until the interrupted routine reaches "the end of the current row" (Third Reiffin affidavit, Par. 16; Third Wadsworth affidavit, Par. 26(4)). Furthermore, Lawrence states that control is passed to the keystroke processor 9 and not directly to the text editor 10 as an interrupt service routine.

Neither the final rejection nor the Examiner's Answer cited a single instance in the Lawrence specification of either the term "interrupt" or the term "return", or indeed anything even remotely equivalent to either.

Only in Claim 4 does the Lawrence patent state:

"...said editing means includes means to interrupt said formatting means upon receipt of a keystroke from said keyboard invoking said editing means to perform a text editing operation."

This limitation is confused, self-contradictory, inconsistent with the specification, and not enabled by any hardware or software disclosed in the patent. It is confused and self-contradictory because it recites in effect that the editing means includes means which invokes the editing means, a nonsensical statement of recursion. Neither the editing means nor any means included therein can invoke anything until it gets control of the central processing unit, and once it gets control there is no longer any point in having it invoke itself.

This claim limitation is inconsistent with the specification statements quoted above. The Examiner's Answer admits in Paragraph 21 that the reference contains "some inconsistency of the terminology, i.e. interrupted versus invoked."

This claim limitation, if interpreted in the manner proposed by the Examiner to mean that the editor is an interrupt service routine which when finished results in an automatic immediate return to the formatter which then resumes formatting at the point in the text where it was interrupted, would result in an inoperative system for the reasons discussed below under point II. The same fatal defect would arise if the interrupt hardware of Catiller were implemented in the Lawrence system. In any event, the Lawrence patent includes no enabling disclosure which would successfully provide the mode of operation contended by the Examiner.

The supposed "inconsistency" conceived by the Examiner is "resolved in favor of 'interrupted'" because:

"... Lawrence et al explicitly discussed the different priorities among different processes wherein keystroke processing had a higher priority than text editing which in turn had a higher priority than interpreting (formatting) ..."

and also because allegedly:

"It was well known to one of ordinary skill in the DP art at the filing date of Lawrence et al reference that interrupts, (not invoke), are normally associated with priority arrangements."

The Examiner's contention is in effect that the mere statement that one process has priority over another process constitutes a disclosure that the first process is an interrupt service routine. If this were true, then each of ten (10) of the twelve (12) processes listed in the priority table in Col. 7, Lines 20-29 of the reference has priority over some other process and therefore must be deemed to be implemented in its own individual interrupt service routine. That is, every process except "monitoring" and "scheduling" would be an interupt service routine!

If so, then "interpreting", one of the processes listed in the table as having priority over some other processes, must also be deemed an interrupt service routine. Hence the inevitable result of the Examiner's flawed logic is that the interpreter/formatter is an interrupt service routine and therefore gets control only temporarily during an interrupt. This would be the very antithesis of appellant's limitation recited in Claim 51 that:

"the code processor program thereafter continuously maintains control of the central processing unit, except during interrupts" (Emphasis added.)

Indeed, if mere listing above the lowest level of the priority table designates a process as an interrupt service routine, then the inevitable result is that every process mentioned throughout the entire reference (except monitoring and scheduling) must be deemed an interrupt service routine since all other processes mentioned in the reference also appear in the priority table. In this event, what process would be left to function as the main routine which is to be interrupted by the listed processes?

Of course, the answer is that the examiner's argument is untenable because priorities may be assigned to processes that are <u>not</u> "associated" with interrupts. For example, the fundamental "if then else" construct in every modern high-level programming language establishes priorities without being "associated" in any way with interrupts. Such a construct, with conventional polling of the character input queue, might be employed to achieve the priorities listed in Lawrence without need for any interrupts. The Examiner's vision of interrupts in the Lawrence priority list is suggested solely by appellant's own disclosure.

In any event, even if the word "interrupt" had been expressly mentioned in the reference, the mere mention of the concept would not constitute an enabling disclosure of an operative system.

The Maddock text editor is neither an interrupt service routine nor does it get control from the background formatter immediately upon striking a key. As stated in the Maddock specification at Col. 5, Lines 46-54:

> "...background formatting...is run in short <u>uninterrupted</u> bursts which, in this embodiment will process one line of text a time. [sic] *** once the background formatting is commenced it will <u>not be interrupted until the reformatting of the current line is completed</u>." (Emphasis added.)

An interrupt service routine gets control immediately and cannot be delayed "until the reformatting of the current line is completed" (Third Reiffin affidavit, Par. 16).

II. THE LAWRENCE ET AL. AND MADDOCK FORMATTERS DO NOT REGAIN CONTROL UPON RETURN FROM AN INTERRUPT AND DO NOT RESUME PROCESSING WHERE THEY WERE INTERRUPTED

The Lawrence et al. formatter neither regains control of the central processing unit upon return from a text editor service routine interrupt nor resumes processing the text at the location in the text store where it was allegedly "interrupted". Instead, the Lawrence formatter always resumes formatting "at the top of the screen" (Col. 10, Line 68; Col. 11, Rows 5,12), except when moving a screen forward it starts "at the second row of the screen" (Col. 10, Rows 56,57).

Appellant's Claim 51 recites:

"said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key, and having a return instruction at the end thereof,

said central processing unit having means responsive to execution of said return instruction to pop from the stack into the program counter the memory address of the code processor program instruction immediately following the instruction interrupted by striking said key, whereby the code processor program continues processing said character codes at the location in the buffer where the code processor program was interrupted."

Appellant's Claim 54 recites:

"means thereafter to return control of the central processing unit to the code processor program to enable the latter to resume processing said character codes at the location in the buffer where the code processor program was interrupted."

The above claim limitations provide one of the major advantages of appellant's invention. That is, the code processor does not have to start all over again as each new character or line of code is entered at the keyboard by the operator, and is able to resume processing at the point where it was previously interrupted.

This mode of operation would be pointless in the Lawrence formatter because the formatting of a screenful of characters occurs almost instantly in a time interval of a fraction of a second. Nothing is lost by starting "at the top of the screen" at every invocation of the Lawrence formatter. Nothing would be gained by having the Lawrence formatter resume formatting at the point where it was previously interrupted; that is, the time saved would be insignificant compared to the added complexity that would be involved in such a modification of the reference.

However, in appellant's language processor, such as a compiler, the processing of a program or other language code may take many minutes or even the better part of an hour, and a primary object of appellant's invention is to avoid the tedium and wasted time inherent in conventional compilation and similar language processing techniques, as described on Pages 1,2 of appellant's specification. This object is achieved by having the processor operate concurrently as the code is entered at the keyboard and resume after each keystroke at the very point in the code where it was previously interrupted by the keystroke, so that the processor keeps up with the entry of code by the operator and completes its processing of the entire program or other lengthy language code soon after the last line of code has been typed into the system.

Not only would this mode of operation be pointless in the

Lawrence system, it would also be impossible. The Lawrence system must resume formatting "at the top of the screen" as stated in the reference specification because otherwise the Lawrence system would have no way of preventing the formatter from resuming at a point in the text subsequent to a line which was just changed and which therefore requires formatting. That is, Lawrence has nothing which would prevent the formatter from resuming its formatting function at a row in the text store which is subsequent to the row at which the text editor has just made a change in the text.

For example, if the Lawrence text editor were to interrupt to make a change in the tenth row of the text at the instant when the formatter was processing the twentieth row, upon return from the text editor's proposed interrupt the formatter would resume processing at the twentieth row and the change in the tenth row would not be formatted. Appellant's invention obviates this fatal defect in the novel manner described at Pages 20-22 of his specification.

Therefore, the Examiner's proposed modification of Lawrence to include the Catiller interrupt hardware, for enabling control to be automatically returned to the formatter at the point of interruption, would render the Lawrence system inoperative.

Of course, these considerations arise only if the Examiner is correct that the Lawrence text editor obtains control as an interrupt service routine and that the Lawrence interpreter/formatter is interrupted rather than "terminated" as disclosed at Col. 11, Line 52. It was argued above under point I. that the Examiner 's interpretation of the disclosure in this regard is contrary to several explicit statements in the Lawrence specification and abstract, and furthermore the Examiner's interpretation is unsupported by a single statement in either the specification or abstract.

The Madddock background reformatter is apparently regarded by the Examiner as the equivalent of appellant's code processor program. This reformatter does not regain control of the central processing unit upon return from a text editor interrupt service routine, nor does it resume processing the text at the location in the buffer where it was previously interrupted. As stated in the Maddock specification at Col. 4, Lines 60-66 and Col. 5, Lines 6-9:

"...updating or reformatting of the document following editing to be done ... as a background process run in individual bursts _when there are no keystrokes pending in the keystroke queue_.

***

"Reformatting of the document following the editing operation is performed subsequently _when there are no keystrokes pending in the keystroke queue_." (Emphasis added.)

III. THE LAWRENCE ET AL. AND MADDOCK FORMATTERS DO NOT CONTINUOUSLY MAINTAIN CONTROL OF THE CENTRAL PROCESSING UNIT

The Lawrence interpreter/formatter does not continuuously maintain control of the central processing unit, except during interrupts.

Appellant's Claims 51 and 54 each recite:

"means to cause the central processing unit to execute said code processsor program instructions whereby the code processor program <u>continuously maintains control of the central processing unit, except during interrupts</u>, so as to process the code concurrently as the code is being entered into the system,"

The Lawrence formatter gets control only when it is "invoked" or "re-invoked" by the text editor (Third Reiffin affidavit, Par. 17,18; Third Wadsworth affidavit, Par. 26), and it maintains this control only for the very brief instants that it takes to perform its functions of determining the cursor position and formatting a screenful of rows. Indeed, the ideal screen formatter performs its function in a fraction of second so as to appear instantaneous to the eye. If the formatter maintained control it would <u>not</u> be necessary for the text editor to "re-invoke" it after each "operation on the text is performed". As stated in the Lawrence specification at Col. 3, Lines 30-38:

"The text editor 10 employs a two-step process. Firstly, it <u>invokes</u> the interpreter/formatter 6 to determine the physical cursor position relative to the text in the store 2. At the end of this phase, a corresponding logical cursor position is known within the store. Secondly, the particular operation on the text is performed (insertion/deletion) at the logical position and the interpreter/formatter 6 is <u>re-invoked</u> to display the effect of the change on the screen." (Emphasis added.)

As stated in Paragraph 26 of the Examiner's Answer in discussing the Maddock reference:

"...the formatting time for a line was short compared to typical inter-keystroke time".

Therefore it would be pointless to have the Lawrence formatter maintain control after the brief instant of screen formatting. It would literally have nothing to do for almost the entire time of this period of alleged control (Third Wadsworth affidavit, Par. 28).

Furthermore, the Lawrence specification at Col. 12, Lines 16-32 expressly states that "Row formatting is terminated when any of the following conditions are detected:", and then lists nine (9) conditions. Not one of these conditions is an interrupt. Therefore the Lawrence formatter does not maintain control "except during interrupts" as recited in appellant's Claims 51 and 54.

Paragraph 58 of the Examiner's Answer asserts:

"...in Lawrence et al system, once the formatting started, the formatter continuously maintained control of the system <u>until it was interrupted</u>." (Emphasis added.)

Not only is there nothing in the Lawrence patent which states or even remotely implies this, but the assertion is contrary to the express disclosure in Col. 12, Lines 16-32 of the nine non-interrupt conditions which terminate the formatter.

The Examiner neither cites anything in the reference to support his assertion, nor gives any reason why the formatter should continue to maintain control long after it has finished formatting the screen, particularly in view of the relatively short formatting time compared to the inter-keystroke time.

Control by the formatter of the central processing unit means that the latter unit is executing the instructions constituting the formatter. What formatter instructions are allegedly being executed by the central processing unit during this waiting period after the formatting operation has finished? Why are they being executed? What earthly purpose might be served by this pointless waiting period, except to create a fictitious mode of operation allegedly anticipatory of the claims in issue? In any event, this fiction is refuted by the express disclosure in Col. 12, Lines 16-32 of the nine non-interrupt conditions which terminate the formatter.

The Maddock background reformatter does not continuously maintain control of the central processing unit, except during interrupts. As stated at Col. 4, Lines 64-65 and Col. 5, Lines 48-49 of the Maddock specification, the background reformatting is run only in short "bursts".

The mere fact that Maddock loosely and inaccurately characterizes this discontinuous sequence of bursts as "a substantially continuous operation" in Col. 5, Lines 63-64 does not anticipate the subject limitation in appellant's claims wherein control is maintained "continuously" in the true and accurate sense of the word.

Furthermore, when the fraction of a second required for reformatting the screen is over and the reformatting operation is finished, does the Examiner contend that thereafter the Maddock reformatter continues to maintain control of the central processing unit? This would be pointless, for the reasons discussed above with respect to Lawrence, and is not disclosed in the Maddock patent.

IV. THE BELATED LAWRENCE CLAIM LIMITATION WAS NOT IN THE ORIGINAL APPLICATION

The belated limitation of the Lawrence patent Claim 1 reads:

"means in said editing means for invoking said formatting means when a text editing <u>step</u> has been completed". (Emphasis added.)

Paragraph 49 of the Examiner's Answer erroneously contends that this limitation "was in original claim 9". However, the Examiner overlooks that the Claim 9 limitation recited the term "process" instead of "step". This is a critical difference in view of the issue of when and how control is passed to the Lawrence formatter.

Webster's New World Dictionary, published in 1980 by Simon and Schuster, Page 1133, includes several definitions of "process", the most pertinent being:

"a particular method of doing something, generally involving a <u>number of steps</u> or operations". (Emphasis added.)

The term "process" in the relied upon limitation of the Lawrence application Claim 9 indicates that control is passed to the formatter after a "number of steps" has been completed, and therefore refutes rather than supports the Examiner's contention that control is passed to the formatter "after each keystroke".

Therefore, contrary to the Examiner's assertions in Paragraphs 46 and 47 of the Examiner's Answer that appellant's allegation is "misleading", it is the Examiner's allegation which is less than accurate. It is ironic that this belated claim limitation also refutes the Examiner's preference for "interrupt" which is admittedly inconsistent with the disclosed "invoke".

V. APPELLANT'S COMPILER IS DISCLOSED IN COMPLETE AND SPECIFIC DETAIL

Paragraph 39 of the Examiner's Answer asserts that appellant failed to disclose his "code processor program (compiler)" and that "there is no software listing in the parent application".

These assertions are utterly inaccurate. It appears that the Examiner has either failed to read or failed to recall appellant's disclosure of the modified Wirth compiler in appellant's present specification and in his parent application Serial No. 425,612. The disclosure is identical in both applications. As stated at Page 17 in the present specification and at Page 14 of the parent specification:

"For clarity in illustration it will be shown how the simple and widely-published compiler PL/O of Prof. N. Wirth (Ref. 12) may be modified for implementation in the present invention."

Reference 12 is listed on Page 28 of the present specification and on Page 22 of the parent specification as:

"12. N. Wirth, 'Algorithms + Data Structures = Programs', Ch. 5, pp. 280-347, 1976, Prentice-Hall, Inc."

This treatise by Prof. Wirth is one of the most well-known classics in computer science, and the author is the reknowned creator of the Pascal language. A complete listing of the PL/O program, copied from the treatise, was filed in appellant's parent application. This published listing includes each and every statement and declaration constituting Prof. Wirth's PL/O compiler.

Pages 17-19 of appellant's present application and Pages 14-16 of his parent specification then disclose in specific detail each and every change to be made in the published Wirth compiler for implementation in the present invention.

Appellant's editor is not disclosed with quite the specificity and detail as the compiler because the editor is simpler and has fewer novel aspects peculiar to the present invention. Nevertheless, every such novel aspect, including each routine and its function, is clearly disclosed in the two specifications in conjunction with appellant's Figs. 5 and 6 of the respective sets of drawings of both applications.

Appellant's disclosure of each and every statement and declaration of the compiler, and of each and every novel routine and function of the editor, is a far cry from the uninformative box 38 labeled "(Text Code)" in Fig. 5 of the Lawrence patent.

VI. THE EXAMINER HAS MISINTERPRETED AND DISTORTED THE AVERMENTS OF THE WADSWORTH AFFFIDAVIT

The Examiner's Answer misinterprets and distorts the averments of the third Wadsworth affidavit. In Paragraph 41 of the Answer the Examiner advances this remarkable <u>non sequitur</u>:

"Affiant clearly demonstrated that he understood the operations of the formatter and editor. If affiant understood the operations, it is reasonable to conclude that affiant, one ordinary skilled in the art, can make the formatter and editor <u>without</u> undue experimentation. In other words, the affidavit of record contradicts appellant's allegation that one skilled in the art can not make the formatter and editor in question."

The affiant "understood the operations" in the sense that the meanings of the English words were comprehended, not in the sense that the implementation of the operations was obvious. According to the Examiner's faulty logic, if an affiant understands the meaning of the operation "fly to Jupiter faster than the speed of light", then <u>ipso facto</u> "it is reasonable to conclude that affiant can make a rocket ship to perform this flight without undue experimentation".

Armed with this fallacious logic, the Examiner in Paragraph 42 inaccurately characterizes the Wadsworth averments as an "implicit admission". What Wadsworth really alleged in Paragraphs 25-29 of his third affidavit is more accurately summarized as follows:

(1) None of the four modes of operation alleged by the Examiner to be performed by the Lawrence system is in fact disclosed in the reference patent (Par. 25-26).

(2) Modification of the Lawrence system to provide these four modes of operation would not be obvious to one skilled in the art (Par. 27,29).

(3) These alleged modes of operation would serve no useful purpose in the Lawrence environment (Par. 28).

(4) Affiant is unaware of anything in the prior art which would suggest these modes of operation to one skilled in the art (second Par. 28).

(5) Neither hardware nor software for implementing the purported functions of the Lawrence "editor" is disclosed in the patent (Par. 29). (In Paragraph 13 Wadsworth referred to the whole Lawrence system as "an editor and only an editor".)

VII. NEW DEPENDENT CLAIMS 57-67 FURTHER DISTINGUISH APPELLANT'S INVENTION

New dependent Claims 57-67 further distinguish appellant's invention in that they add limitations to the effect that the code processor program is a language processor comprising means for analyzing the character codes in the buffer for conformity with the rules of the language. This is entirely alien to the screen formatters of Lawrence et al. and Maddock. As averred in Paragraph 14 of the third Wadsworth affidavit:

"The disclosed Lawrence system...performs neither lexical analysis nor syntactic analysis nor semantic analysis of the text symbols...It knows and cares nothing about the syntax or meaning of the text symbols, but merely loads them in the display buffer for display upon the screen. The text symbols may constitute utter garbage but this will have no affect upon the operation of the Lawrence interpreter/formatter."

CONCLUSION

It is therefore respectfully submitted that:

(1) The Lawrence et al. reference does not contain an enabling disclosure of the interactions and modes of operation relied upon in the final rejection and in the Examiner's Answer;

(2) Said interactions and modes of operation are undisclosed in and inconsistent with the actual disclosures of the Lawrence et al. and Maddock references;

(3) The Examiner's proposed modification of Lawrence et al. in view of Catiller would result in an inoperative system, would be pointless, and therefore would not be obvious to one skilled in the art;

(4) Appellant's Claims 51, 54 and 56 to 67 inclusive are patentable over the references; and

(5) The final rejection is untenable and should be reversed.

Respectfully submitted,

January 19, 1989.

Martin G. Reiffin

Martin G. Reiffin,
Appellant
5439 Blackhawk Drive
Danville, CA 94526
(415) 838-6980